Shrinking Raspberry Pi SD Card Images

***Update* 2017-02-02: A command-line utility to perform all of these stages automatically, has now been posted on my Github repository.**

I teach coding with the Raspberry Pi computer at Cotswold Raspberry Jam, and as part of that I often need to store assorted SD card images, either for backups, or as master versions which I copy out to the growing number of tutorial machines that we have.

These images can be quite large. Shrinking these images has a number of benefits, including:

- Takes us less storage space (which can really matter on a cheap SSD on an old cheap netbook!)
- Takes less time to transfer over the network
- Takes a LOT less time to write to SD cards

Throughout this article I'm going to use the initialism "SD" to refer to both full-size Secure Digital cards, and Micro SD cards

Remember, the Raspberry Pi's Raspbian operating system comes with a tool to resize the filesystem to the largest size the SD card will support (`sudo rpi-config` , then select Expand Filesystem). So you really do lose nothing by shrinking the image, because it's easy to expand it back again.

CAUTION: WRITING DISK IMAGES INCORRECTLY CAN NEAR-PERMANENTLY RENDER YOUR COMPUTER INOPERABLE. Follow the instructions carefully and note any warnings. If you're not sure, stop here.

Use a Linux PC with a hard disk or SSD

To do this properly, you need a Linux PC with a proper hard disk or reasonably-sized SSD. You *can* do this on a Raspberry Pi using an external USB card reader, but it'll be monumentally slow because you'll be limited to the USB2 speed of both the external card reader and the internal SD card slot. If you really must do this entirely on a Raspberry Pi, try to use SD cards that have minimum read and write speeds at or above 25 megabits per second (which is the maximum that USB2 supports).

This article assumes you are using a Debian-based Linux distribution; I used Linux Mint 17.2 but any vaguely recent Debian-based distro such as Ubuntu or, yes, Raspbian should work fine.

You will also need to be running a desktop. We're going to use `gparted` which is a graphical partition editing tool. There is a commandline alternative, but it makes things far more difficult than this tutorial already is.

Before you start - use Raspbian not NOOBS

First off, if your goal is to have a small image file, then you want to be working from an SD card that was imaged only with the Raspbian operating system. It's all very well to use NOOBS - it's great for beginners - but it wastes quite a lot of space. If you want to shrink card images then you are not a beginner so NOOBS is no longer for you. NOOBS cards also have a number of different partitions which are difficult to work with. If your goal is a small card image... don't start with NOOBS. Image a new card using the lastest Raspbian-only image from raspberrypi.org/downloads and then copy over whatever you need from your NOOBS card.

Do not proceed any further with this how-to article if you are still using NOOBS. I realise this is an "I wouldn't start from here" instruction, but that's the practicality of the situation. You could use your favourite search engine to search for "zero fill" techniques which will make a NOOBS image compress better, but it won't shrink the image file itself.

The Rasbian operating system has become quite large recently (Oct 2015). The Libreoffice and Wolfram packages are particularly large, about half a gig each. If you don't intend to use these, boot up your Raspberry Pi, go to a terminal session and type:

```
sudo apt-get remove --purge libreoffice-*
sudo apt-get remove --purge wolfram-engine
```

You can always add them back later with `sudo apt-get install libreoffice wolfram-engine` .

You can also clear out the cache of downloaded .deb packages. This might save you several hundred megabytes if you've done a lot of updates or installed a lot of extra packages:

```
sudo apt-get clean
```

Don't forget to properly shut down your Raspberry Pi (for example, with `sudo shutdown -h now`) before removing the SD card.

Making the image

Let's start by making an image. Pop your SD card in your PC's card reader. Most likely the file manager will open a window or two. Close these windows if so.

Now find out where the card is mounted. Go to a terminal session and type:

```
df -h
```

You'll see something like:

```
/dev/sde1                56M   20M   37M  36% /media/aoakley/boot1
/dev/sde2               7.2G  3.8G  3.0G  56% /media/aoakley/ec2aa3d2-eee7-454e-8260-
d145df5ddcba
```

The important thing is that you now know that your SD card is on /dev/sde . It has two partitions, /dev/sde1 and /dev/sde2.

Your system might mount the card somewhere else, such as /dev/sdg or even /dev/sdb. Make a note of where your card is mounted and use this wherever I use /dev/sde , /dev/sde1 or /dev/sde2.

Let's unmount this but leave the card in the card reader. This will let us take an image.

```
sudo umount /dev/sde1 /dev/sde2
```

Obviously if yours is on /dev/sdb or /dev/sdg or whatever, you'll need to type that in appropriately. It is highly unlikely that your SD card is on /dev/sda unless you're mad enough to do this entirely on a Raspberry Pi. Typically /dev/sda will be your hard disk or SSD, so don't mess with that.

`sudo` might ask for a password. Check that you really, really haven't typed /dev/sda unless you really, really know what you're doing, then enter your password.

I use `dcfldd` for making card images, which is a replacement for the old `dd` disk duplication program. `dcfldd` has a number of improvements, most notably a progress meter so you can see it working, and be confident that it hasn't crashed.

Install `dcfldd` if you haven't already got it (it won't hurt to try to install it again):

```
sudo apt-get update && sudo apt-get install dcfldd
```

Now take the image. Again, change /dev/sde to wherever your SD card is mounted.

```
sudo dcfldd if=/dev/sde of=imagename.img
```

You can change imagename to whatever you like - I use YYYYMMDD dates and names, so 20151009-tutorial.img for example.

The image will start to be taken, together with a nice progress counter (which you don't get with old-style `dd`). When it finishes there may be a pause before you get the command line prompt back - give it a minute or two to flush the cache.

Once finished, you should force a synchronise of any outstanding input or output (there shouldn't be any, but just to be

sure), then the card will be safe to remove.

```
sudo sync
```

You may now safely remove the card.

Next, let's change the ownership of the .img file. The image file will be owned by root (because we used sudo). It's probably a wise move to change the ownership to your user. For example, my username is aoakley but you'll need to change this to your username:

```
sudo chown aoakley.aoakley imagename.img
```

We use aoakley twice because we are changing both the ownership and the group.

Okay, you've got an image file; you have backed up your SD card. But it's probably quite a big file - as big as the SD card itself. Let's start making it smaller.

Resizing a partition within an image file

For this, we're going to use gparted. If you don't have gparted installed:

```
sudo apt-get update && sudo apt-get install gparted
```

Normally gparted can only edit physical storage. We're going to do a little magic with "the loopback device" to make the PC think that the image file is mounted as a real SD card.

The Raspbian operating system has two partitions. 1 is the boot partition, which is tiny and doesn't need shrinking. Partition 2 is where everything else is stored, and typically has lots of free space. Let's have a look at these partitions:

`sudo fdisk -l imagename.img` This should show something like:

```
        Device Boot      Start         End      Blocks   Id  System
imagename.img1            8192      122879       57344    c  W95 FAT32 (LBA)
imagename.img2          122880    15415295     7646208   83  Linux
```

Take a note of the START sector for the second partition. In the above example, this is 122880. Write it down because we're going to use it later on in this tutorial, as well as right now. Let's mount that partition:

```
sudo losetup /dev/loop0 imagename.img -o $((START*512))
```

Replace the word START with the start sector number of your second partition; in my case, 122880. If you get a message that the device is busy, this is probably because you've previously instanced it incorrectly; remove the existing loop0 with `sudo losetup -d /dev/loop0` and try again.

By default, gparted won't read loopback devices, so we need to start it with the loopback parameter:

```
sudo gparted /dev/loop0
```

gparted should start in a desktop window and show the second partition. Click the /dev/loop0 partition and select Partition menu, Resize/Move . Change the value of "New Size" so that it is slightly above "Minimum Size". I suggest allowing 20MB extra space. Click the Resize/Move button when done.

Now click Edit menu, Apply All Operations. The data will be moved to fit into the new size.

When complete, it will display the new size. Make sure you **note down the new size** before you exit.

If the size is not displayed, click the triangle icon next to Details, and the triangle icons that appear nested below them, until you can see the new size. Eventually you'll see a line like **"resize2fs -p /dev/loop0 1410048K"** where the number in K is

the new size in kilobytes.

No, really, **note down the new size** before you exit.

Now remove the loopback device for the second partition, create a new loopback device for the whole image and edit the partition table to reflect the new smaller size:

```
sudo losetup -d /dev/loop0
sudo losetup /dev/loop0 imagename.img
sudo fdisk /dev/loop0
```

`fdisk` is rather basic to use.

- Enter `d 2` to delete the table entry for the second partition
- Enter `n p 2` to create a new second partition entry
- Enter the START sector number that you used earlier, as the start sector. In my example it was 122880.
- Enter +NEWSIZE as the new size. **Don't forget the plus at the start.** This is the new size that you noted down before exiting gparted. If your number was in K (kilobytes) or M (megabytes) then type that letter in too (for example +1410048K).
- Enter `w` to write the new partition table and exit

You may see a message telling you that the new partition table can't be used until the next reboot - don't worry about this message; it doesn't really apply to the loopback devices which we're creating and destroying.

That's the partition resized, and the partition table updated. Now we can remove the loopback device, then we just need to trim the empty space from the end of the image file. Let's look at the new partition table and then destroy the loopback device:

```
sudo fdisk -l /dev/loop0
sudo losetup -d /dev/loop0
```

Note down the END sector of the second partition. In my case this is 8615936. Now let's trim down the file to this length, replacing END with your end sector number:

```
truncate -s $(((END+1)*512)) imagename.img
```

And you're done. For extra marks, you can fill any empty space with zeroes to make it slightly better to compress and a teeny, tiny bit faster to write to in use:

```
sudo losetup /dev/loop0 imagename.img -o $((START*512))
sudo mkdir -p /mnt/imageroot
sudo mount /dev/loop0 /mnt/imageroot
sudo dcfldd if=/dev/zero of=/mnt/imageroot/zero.txt
sudo rm /mnt/imageroot/zero.txt
sudo umount /mnt/imageroot
sudo rmdir /mnt/imageroot
sudo losetup -d /dev/loop0
```

If you're keeping the image for backup or archival purposes, you can now compress the file with: `zip imagename.zip imagename.img`

...or use `tar cvzf` or whatever your favourite file compression tool is. Don't forget to uncompress it (for example, `unzip` or `tar xvzf`) before trying to write it to an SD card.

Copying the image to a fresh SD card

Once you've got your shrunk .img file, you can copy it to another SD card. Insert your blank card, close any popup file

```
sudo umount /dev/sde1 /dev/sde2
sudo dcfldd if=imagename.img of=/dev/sde
```
manager windows, unmount it and write the image: `sudo sync`

Remember you may need to change /dev/sde , /dev/sde1 and /dev/sde2 depending where your system mounts your cards.

When you use the new card for the first time, remember to expand the filesystem to fill the space on the new card. This ther requires a reboot. From the Raspberry Pi's terminal, do `sudo raspi-config` , then select Expand Filesystem. Alternatively you can do this directly from the command line with:

```
sudo raspi-config --expand-rootfs
sudo shutdown -r now
```


Top - More Computing Articles - Article Index - aoakley.com